Configuring Data Augmentations to Reduce Variance Shift in Positional Embedding of Vision Transformers

Bum Jun Kim, Sang Woo Kim*

Department of Electrical Engineering, Pohang University of Science and Technology
{kmbmjn, swkim}@postech.edu

Abstract

Vision transformers (ViTs) have demonstrated remarkable performance in a variety of vision tasks. Despite their promising capabilities, training a ViT requires a large amount of diverse data. Several studies empirically found that using rich data augmentations, such as Mixup, Cutmix, and random erasing, is critical to the successful training of ViTs. Now, the use of rich data augmentations has become a standard practice in the current state. However, we report a vulnerability to this practice: Certain data augmentations such as Mixup cause a variance shift in the positional embedding of ViT, which has been a hidden factor that degrades the performance of ViT during the test phase. We claim that achieving a stable effect from positional embedding requires a specific condition on the image, which is often broken for the current data augmentation methods. We provide a detailed analysis of this problem as well as the correct configuration for these data augmentations to remove the side effects of variance shift. Experiments showed that adopting our guidelines improves the performance of ViTs compared with the current configuration of data augmentations.

Introduction

Vision transformers (ViTs) (Dosovitskiy et al. 2021) have demonstrated remarkable performance in a wide range of vision tasks. They have replaced the dominance of previous convolutional neural networks (CNNs) (Simonyan and Zisserman 2015; He et al. 2016), exhibiting improved modeling ability. Now, ViTs have become the standard backbone in image classification as well as other downstream tasks, such as semantic segmentation.

Compared with CNNs, training ViT requires a much larger or more diverse dataset. The original study of ViT (Dosovitskiy et al. 2021) observed that directly training ViTs on ImageNet[1] dataset exhibits worse results than CNNs. To overcome this problem, they pretrained ViTs on the JFT-300M dataset, which is their in-house dataset with a larger size. Subsequently, they fine-tuned the ViTs on the ImageNet dataset, which then showed improved performance compared with CNNs. Afterwards, other follow-up studies (Touvron et al. 2021a; Touvron, Cord, and Jégou 2022) attempted to train ViTs using only ImageNet and found that successful training of ViTs requires rich data augmentations, including Mixup (Zhang et al. 2018), Cutmix (Yun et al. 2019), and random erasing (Zhong et al. 2020). Since then, using a larger dataset or rich data augmentations has been discussed as a critical factor in training ViTs to achieve successful performance.

Note that, in fact, these data augmentations were developed during the CNN era. Although Mixup, Cutmix, and random erasing provide rich transformations of data with increased diversity, we should rigorously examine their validity for training ViTs. Eventually, we report a vulnerability for this issue: Certain data augmentations cause a variance shift in the positional embedding of ViT, which leads to inconsistent behavior of the positional embedding and degraded performance of ViT. We discover that this vulnerability is incurred by the distinct architecture in the early stage of ViT, which requires specific conditions for the input image.

In this regard, we inspect whether current data augmentations satisfy this condition (Table 1). We provide a detailed theoretical description of this vulnerability as well as its root cause. Our analysis discovers that Cutmix is safe from this vulnerability, whereas Mixup incurs variance shifts. We also explore other data augmentations such as random erasing and random resize crop, which require specific configurations for the correct use on ViTs. We empirically examined the validity of our guidelines for configuring data augmentations and observed that following them improves the performance of ViTs.

Theory and Practice: Data Augmentations for Vision Transformers

Problem Statement: Variance Shift in Positional Embedding

This study targets ViTs with absolute positional embedding, which is widely used in numerous studies (Touvron et al. 2021b; d'Ascoli et al. 2021; Jiang et al. 2021; Ali et al. 2021; Han et al. 2021; Heo et al. 2021). Although there are several architectures of transformers with relative positional embedding, recent studies such as EVA (Fang et al. 2023) and Hi-

*Corresponding author

[1]In this paper, ImageNet indicates ImageNet-1K (Deng et al. 2009) unless specified otherwise.

Data Augmentation	Image Variance	Our Theory
Upsampling with nearest neighbor	✓ Consistent	Theorem 1
Upsampling with bilinear or bicubic	✗ Inconsistent	Theorem 1
Mixup	✗ Inconsistent	Proposition 1
Cutmix	✓ Consistent	Proposition 2
Random Erasing	▲ Depends on noise	Proposition 3

Table 1: Summary of the findings

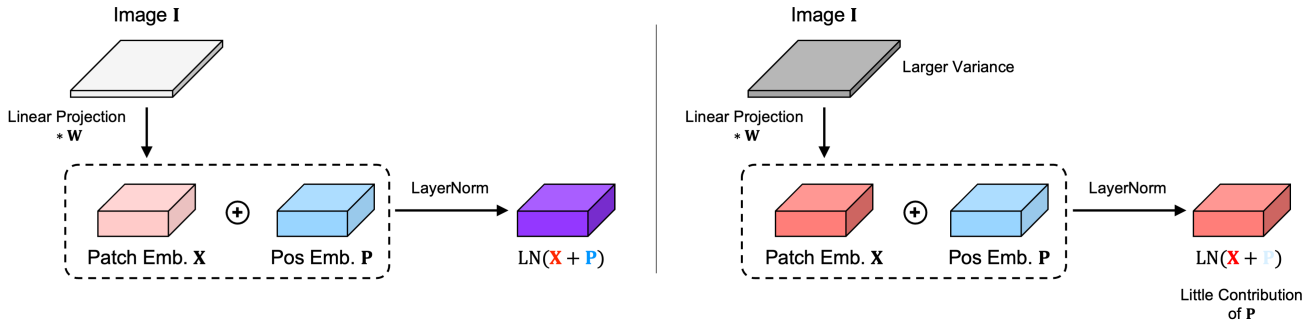

Figure 1: Overview of the early stage of ViT. Variance is depicted by chroma. If the input image or patch embedding exhibits different variances during the training (left) and test (right) phases, positional embedding inconsistently affects the output.

era (Ryali et al. 2023) rather claim that it is desirable to use vanilla ViT without any modification in architecture but with large-scale such as sizes of dataset and model. In consideration of these practices, the absolute positional embedding used in vanilla ViT would still matter in the practice of the current research community; therefore, we focused on this architecture as a scope of this study.

In the early stage, ViT partitions an input image $\mathbf{I}$ into a sequence of patches, which is subsequently modeled by transformer blocks including self-attention operations (Vaswani et al. 2017). The sequence of partitioned patches $\mathbf{X}$—also referred to as patch embedding—is a linear projection of the input image with respect to patch, *i.e.*, $\mathbf{X} = \mathbf{W} * \mathbf{I}$, where $*$ indicates a strided convolutional operation and $\mathbf{W}$ indicates the convolutional kernel. Because the self-attention operation does not recognize positional differences, ViT explicitly employs positional embedding to discriminate between patches at each location. The positional embedding $\mathbf{P}$ is added to the patch embedding and is set to be learnable during training. Now the transformer block starts and applies numerous operations including layer normalization (LN) (Ba, Kiros, and Hinton 2016), multiheaded self-attention, and multilayer perceptron. The first operation of the transformer blocks is LN, which applies normalization using mean and standard deviation to the sum of patch and positional embeddings, *i.e.*, $\mathbf{X} + \mathbf{P}$.

Owing to the first LN, the contribution of positional embedding $\mathbf{P}$ depends on its relative variance with respect to the operand $\mathbf{X}$ (Figure 1). For example, if $\mathrm{Var}[\mathbf{X}] > \mathrm{Var}[\mathbf{P}]$, patch embedding is far more dominant than positional embedding, which decreases the contribution of positional embedding. Specifically, the gradient $\frac{\partial \mathrm{LN}(\mathbf{X}+\mathbf{P})}{\partial P}$ decays with a factor of $\sqrt{\mathrm{Var}[\mathbf{X}]}$, which indicates that the larger variance of the patch embedding reduces the contribution of the positional embedding (Kim et al. 2023). Similarly, a larger variance of the positional embedding decreases the relative contribution of the patch embedding, which is equivalent to reducing the variance of the patch embedding. To ensure a consistent effect from positional embedding, we should avoid inconsistent variance for both patch and positional embeddings.

We extend this rule with respect to the training and test phases: If there is a variance shift in patch or positional embeddings during the training and test phases, ViT exhibits inconsistent behavior. To obtain consistency in the relative contribution of positional embedding, we should ensure a consistent ratio in variance $\mathrm{Var}[\mathbf{X}_{\text{train}}]/\,\mathrm{Var}[\mathbf{P}_{\text{train}}] \approx \mathrm{Var}[\mathbf{X}_{\text{test}}]/\,\mathrm{Var}[\mathbf{P}_{\text{test}}]$ as much as possible. Note that patch embedding is a linear projection of the input image; achieving a consistent variance on patch embedding $\mathrm{Var}[\mathbf{X}]$ is equivalent to ensuring a consistent variance on the input image $\mathrm{Var}[\mathbf{I}]$. Therefore, the term $\mathbf{X}$ in the above ratios can be replaced with $\mathbf{I}$.

However, most data augmentations are only applied during the training phase and turned off during the test phase, which can cause the vulnerability of variance shifts in the input image. In other words, data augmentations do not always guarantee a consistent variance of the input image, which breaks the aforementioned condition. Because modern data augmentations have been developed considering CNNs, which do not employ positional embedding and are free from the variance shift issue, we claim that data augmentations should be configured to be suitable for ViT. Considering this issue, our study investigates the validity of modern data augmentations as well as their correct configu-

rations to avoid variance shifts.

These arguments can be extended with respect to the mean of patch and positional embeddings because LN applies normalization using the mean and standard deviation. Hence, we seek both mean and variance consistency simultaneously. Unless specified otherwise, we mainly examine variance when mean consistency holds (See the Appendix).

Properties of Variance By definition, variance provides a suitably scaled result with respect to its size. Because the variance of a tensor or matrix will be computed on its flattened version, we consider them as vectors. For a vector $\mathbf{v}$, each element v is sampled from the same distribution. By definition, variance is invariant to shuffling the order of elements in a vector. Because variance is scaled with respect to the sample size, duplicating the vector multiple times yields the same variance: $\mathrm{Var}[(\mathbf{v}; \cdots; \mathbf{v})] = \mathrm{Var}[\mathbf{v}]$. This property can be extended to the concatenation of different vectors (See the Appendix). Considering the practical scenario of vision tasks where the vector corresponds to a large feature map, we assume that the size of the vector is sufficiently large. The large number of samples assures that cropping a vector to use a subset of the vector yields approximately the same variance: $\mathrm{Var}[\mathrm{Crop}(\mathbf{v})] \approx \mathrm{Var}[\mathbf{v}]$. Considering the practical scenario, we assume that the vector is not a constant vector.

Variance Shift Due to Upsampling

For an input image with a spatial size of $N_h \times N_w$, using a patch size of N_p yields a patch embedding with a spatial size of $(N_h/N_p) \times (N_w/N_p)$, and the additive positional embedding has the same spatial size of $(N_h/N_p) \times (N_w/N_p)$. Although the common practice for training ViT is to use a fixed size of training image, such as 224×224, the image size may differ during the test phase. The use of an arbitrary size for the test image precludes the addition of positional embedding that has a fixed size. To address this issue, the original ViT study (Dosovitskiy et al. 2021) proposed upsampling the positional embedding, where bicubic interpolation is used with an upsampling rate determined by the size of the input image. Though upsampling the positional embedding might seem like a valid approach to cope with an arbitrary size of image, we claim that upsampling operations can affect variance, which causes the vulnerability of variance shift:

Theorem 1 *Upsampling yields* $\mathrm{E}[\mathrm{UP}(\mathbf{v})] = \mathrm{E}[\mathbf{v}]$ *and* $\mathrm{Var}[\mathrm{UP}(\mathbf{v})] = \mathrm{Var}[\mathbf{v}]$ *for duplication-type upsampling but not for interpolation-type upsampling.*

For example, bicubic or bilinear upsampling corresponds to the interpolation-type, which cannot conserve both mean and variance simultaneously after its upsampling. Duplication-type upsampling indicates repeating exact elements in the vector, where the nearest neighbor upsampling corresponds. Indeed, nearest neighbor upsampling is equivalent to duplicating a vector and shuffling elements into the correct order, which conserves variance. This statement is stronger than Theorem 3.2 of Kim and Kim (2024).

Here, we rewrite our theorem as $\mathrm{Var}[\mathrm{UP}(\mathbf{v})] = k\,\mathrm{Var}[\mathbf{v}]$ where $k = 1$ for duplication-type upsampling and $k \neq 1$ for interpolation-type upsampling. The variance ratio k is a constant that depends on the specific interpolation method (Table 2). The upsampling rate to adjust spatial size is considered to be a real number larger than one, and its choice does not affect the variance ratio k. This statement applies to arbitrary dimensional upsampling: Considering 1D upsampling that yields scaling by k_{1D}, 2D upsampling can be described as a repetition of 1D upsampling for two sides, and thus we obtain $k_{2D} = k_{1D}^2 \neq 1$ for the interpolation-type, and similarly $k_{2D} = k_{1D}^2 = 1$ for the duplication-type. Note that variance itself provides a suitably scaled result with respect to its size, which ensures that different variance is not caused by the increased size from upsampling. See the Appendix for a detailed proof and discussion. We are not saying that nearest neighbor upsampling is superior; rather, our claim is that as long as we follow the current practice of applying bicubic upsampling to positional embedding to match the size, variance exhibits inconsistency. Now, we investigate practical vision tasks where the upsampling operation incurs a variance shift in positional embedding.

Value	Bicubic	Bilinear	Nearest Neighbor
k_{2D}	0.7295	0.3927	1.0000
k_{1D}	0.8541	0.6267	1.0000
$1/\sqrt{k_{2D}}$	1.1708	1.5957	1.0000
$1/\sqrt{k_{1D}}$	1.0820	1.2632	1.0000

Table 2: We empirically measured the variance ratio for different upsampling methods. Bicubic and bilinear upsamplings exhibit $k < 1$, whereas nearest neighbor upsampling yields $k = 1$.

Image Classification Scenario Since the introduction of VGGNet (Simonyan and Zisserman 2015), for training image classification tasks such as ImageNet, `RandomResizeCrop` has been widely deployed. This operation upsamples the training image to a slightly larger size and crops it to a fixed size, such as 224×224 pixels. During the test phase, arbitrary image size $N_h \times N_w$ is allowed, but similarly, using an inference crop percentage p, a resize operation into $N_h/p \times N_w/p$ and a center crop operation into $N_h \times N_w$ are applied in a row. For example, using inference crop percentage 0.9 and a test image size 224×224, it applies a resize operation to 248×248 and a center crop operation to 224×224.

Thus, the image is subjected to resize and crop operations, which decrease variance. However, whether to upsample the positional embedding depends on the size of the image. When positional embedding is not subjected to upsampling during the test phase, no variance shift occurs. The problem arises when using an arbitrary size of a test image, which requires upsampling of both the input image and positional embedding. This scenario breaks the consistency of

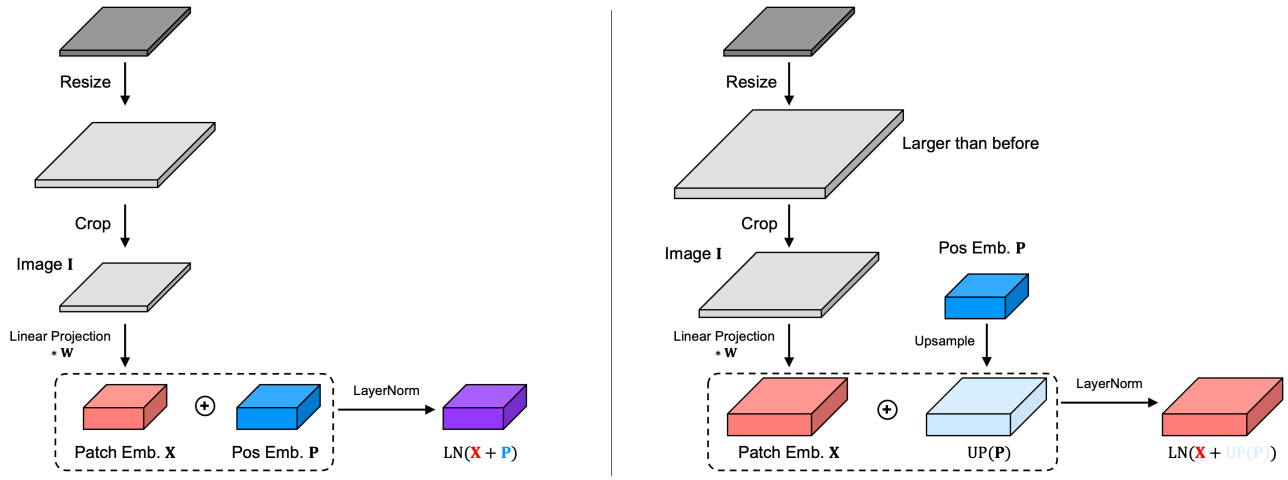

Figure 2: Positional embedding is upsampled depending on the size of patch embedding, which causes inconsistent variance of positional embedding

the variance ratio (Figure 2). Specifically, we obtain

$$\mathrm{Var}[\mathbf{I}_{\text{train}}] = \mathrm{Var}[\mathrm{Crop}(\mathrm{UP}(\mathbf{I}))] = k\,\mathrm{Var}[\mathbf{I}], \quad (1)$$

$$\mathrm{Var}[\mathbf{P}_{\text{train}}] = \mathrm{Var}[\mathbf{P}] = \mathrm{Var}[\mathbf{P}], \quad (2)$$

$$\mathrm{Var}[\mathbf{I}_{\text{test}}] = \mathrm{Var}[\mathrm{Crop}(\mathrm{UP}(\mathbf{I}))] = k\,\mathrm{Var}[\mathbf{I}], \quad (3)$$

$$\mathrm{Var}[\mathbf{P}_{\text{test}}] = \mathrm{Var}[\mathrm{UP}(\mathbf{P})] = k\,\mathrm{Var}[\mathbf{P}], \quad (4)$$

$$\mathrm{Var}[\mathbf{I}_{\text{train}}]/\,\mathrm{Var}[\mathbf{P}_{\text{train}}] = k\,\mathrm{Var}[\mathbf{I}]/\,\mathrm{Var}[\mathbf{P}], \quad (5)$$

$$\mathrm{Var}[\mathbf{I}_{\text{test}}]/\,\mathrm{Var}[\mathbf{P}_{\text{test}}] = \mathrm{Var}[\mathbf{I}]/\,\mathrm{Var}[\mathbf{P}]. \quad (6)$$

Therefore, if positional embedding is subjected to upsampling, its variance varies, whereas the variance of the image—equivalently variance of patch embedding—remains the same. In this scenario, the variance of positional embedding during the test phase is k times of that during the training phase, which we call the variance shift.

To remove the variance shift, we claim that the variance of positional embedding should be calibrated. One may attempt to match variances by replacing bicubic upsampling with nearest neighbor upsampling, which conserves variance. However, we observed that nearest neighbor upsampling produces poor performance because it provides unnatural interpolation that is unsuitable for visual and perceptual tasks. Considering this observation, we should employ natural upsampling such as bicubic upsampling while its variance shift should be removed. Here, we found that simply rescaling the upsampled positional embedding by $1/\sqrt{k}$ during the test phase works suitably in practice. In other words, to remove variance shift in the above scenario, we propose amplifying positional embedding during the test phase by $\mathbf{P}_{\text{test}} = \mathrm{UP}(\mathbf{P})/\sqrt{k}$, whose variance is $\mathrm{Var}[\mathbf{P}]$, ensuring no variance shift.

The proposed rescaling is compatible with the current source code of ViTs and can be implemented by inserting few lines of code during the test phase; moreover, the original ViT model can be used without any retraining. The value $1/\sqrt{k}$ for each upsampling is summarized in Table 2, which does not need hyperparameter tuning. When the rescaled positional embedding is pre-computed and saved in advance, it does not impose additional computational cost in the main inference, which is actually a free gain beyond the trade-off between computational cost and performance.

Semantic Segmentation Scenario We additionally describe the variance shift for a downstream task, specifically in the semantic segmentation scenario. Semantic segmentation is one of the major fields in computer vision and refers to the task of generating a semantic mask that classifies each pixel in an image into a specific category. A semantic segmentation network with an encoder-decoder architecture extracts segmentation output that has the same size as the input image, which enables the use of an arbitrary size of the input image. For semantic segmentation on the ADE20K dataset (Zhou et al. 2017) as an example, the common pipeline for data augmentation includes `RandomResize` with a scale (2048, 512) and `RandomCrop` with a size 512×512 during the training phase, whereas it applies `Resize` with a scale (2048, 512) without cropping during the test phase. The resize scale (2048, 512) indicates an upsampling where the maximum edge is no longer than 2048 and the shorter edge is no longer than 512. This behavior yields 1D or 2D upsampling depending on the dataset.

We interpret this practice as follows. Upsampling is applied to images during both the training and test phases, whereas cropping is only applied during the training phase, yielding $\mathrm{Var}[\mathbf{I}_{\text{test}}] = \mathrm{Var}[\mathrm{UP}(\mathbf{I})] = k\,\mathrm{Var}[\mathbf{I}]$. In this case, the input image to ViT has a larger spatial size during the test phase compared with the training phase. To cope with the larger spatial size of the image during the test phase, bicubic upsampling is applied to positional embedding only during the test phase. Therefore, semantic segmentation with ViT yields the same equations as Eqs. 1 to 6 except for the minor difference Eq. 3, and similarly, we propose applying the $1/\sqrt{k}$ rescaling to positional embedding.

In summary, the use of different sizes of input image during training and test phases is prevalent in current practices of vision tasks, such as image classification and semantic

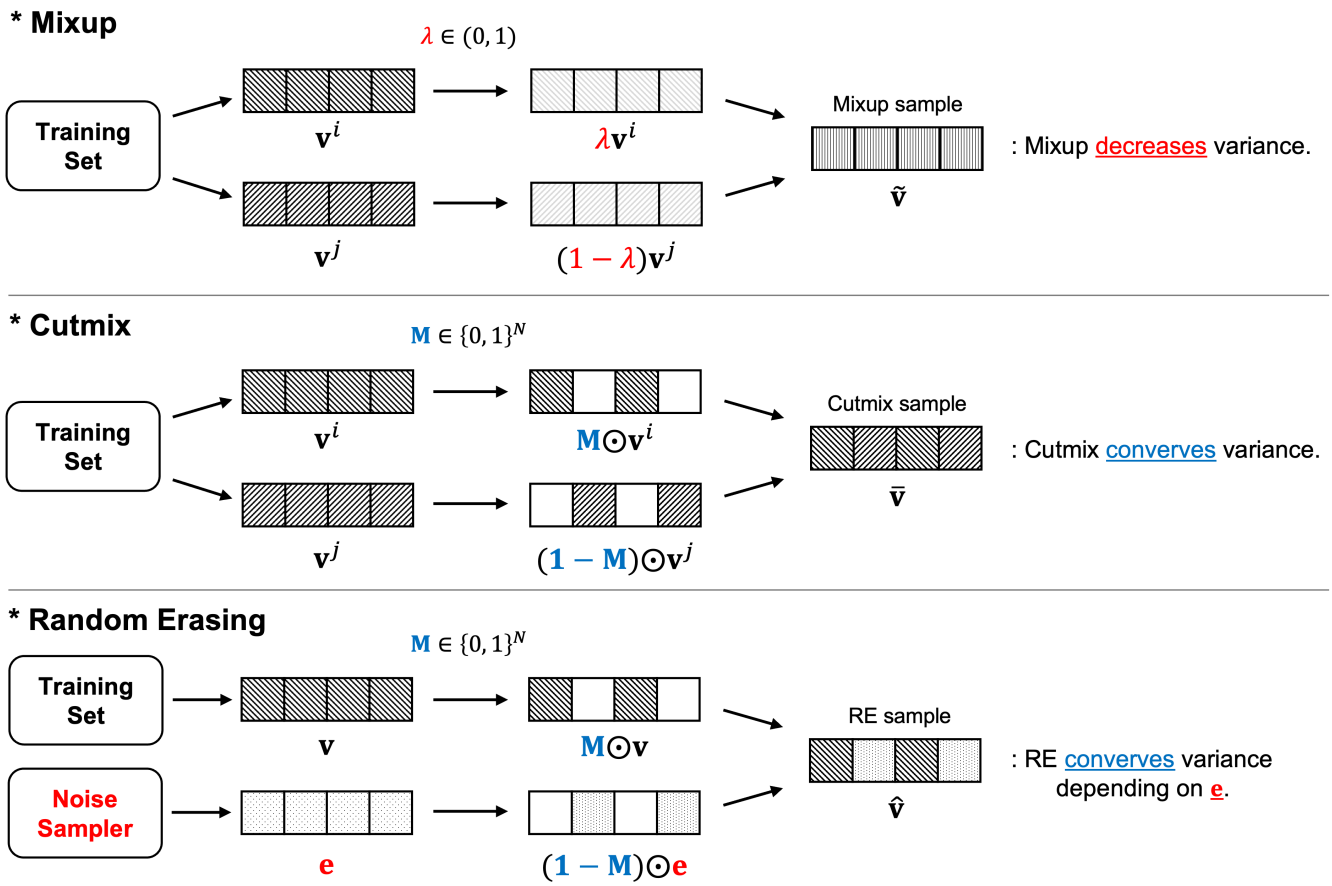

Figure 3: Illustration of Mixup, Cutmix, and random erasing. The level of darkness indicates variance.

segmentation. These practices cause inconsistent application of upsampling on positional embedding during training and test phases, yielding its variance shift. To avoid this variant shift problem, we claim to apply the $1/\sqrt{k}$ rescaling to positional embedding.

Variance Shift Due to Mixup

For image classification tasks, several advanced data augmentations, such as Mixup (Zhang et al. 2018) and Cutmix (Yun et al. 2019), have been proposed. They proposed using a combination of two images for training, which substantially boosted the performance of CNNs. These data augmentations have also been adopted for training ViTs. Indeed, several studies on ViT have revealed their training recipes, including data augmentations and exact hyperparameters (Touvron et al. 2021a; Touvron, Cord, and Jégou 2022; Bao et al. 2022; He et al. 2022; Liu et al. 2021, 2022; Chen, Fan, and Panda 2021). In their training recipes, Cutmix has been steadily preferred. Mixup, however, has been used with a Mixup ratio set to 0.8 to have a slight effect while avoiding a Mixup ratio of 0.5, which corresponds to its maximum usage. Furthermore, the study of EVA (Fang et al. 2023) reported that they turned off Mixup in their training recipe. These practices are explainable through our analysis:

We claim that Mixup causes variance shift, whereas Cutmix conserves variance (Figure 3).

Let $\mathbf{v}^i$ and $\mathbf{v}^j$ be vectors sampled from the training distribution, where $\mathrm{E}[\mathbf{v}^i] = \mathrm{E}[\mathbf{v}^j]$ and $\mathrm{Var}[\mathbf{v}^i] = \mathrm{Var}[\mathbf{v}^j]$. Mixup proposes training a neural network using a Mixup sample $\tilde{\mathbf{v}} = \lambda\mathbf{v}^i + (1-\lambda)\mathbf{v}^j$ with a Mixup ratio $\lambda \in (0,1)$. For the Mixup scheme, we obtain the following property:

Proposition 1 *Mixup decreases variance:* $\mathrm{Var}[\tilde{\mathbf{v}}] < \mathrm{Var}[\mathbf{v}]$.

The proof is straightforward because $\lambda^2 + (1-\lambda)^2 \neq 1$ unless $\lambda = 0$ or $\lambda = 1$, *i.e.*, as long as it is a valid Mixup sample. The inconsistent variance can be observed for other λ beyond $(0,1)$ and for other variants of Mixup (See the Appendix).

In contrast, Cutmix combines two samples using a binary mask $\mathbf{M}$ where each element is either zero or one. Cutmix proposes training a neural network using a Cutmix sample $\bar{\mathbf{v}} = \mathbf{M} \odot \mathbf{v}^i + (\mathbf{1}-\mathbf{M}) \odot \mathbf{v}^j$, where $\odot$ indicates element-wise multiplication. For the Cutmix scheme, we obtain the following property:

Proposition 2 *Cutmix conserves variance:* $\mathrm{Var}[\bar{\mathbf{v}}] = \mathrm{Var}[\mathbf{v}]$.

Because variance is invariant to the shuffling order of a vector, without loss of generality, we may place mask elements

of one at the front and zero at the back. This permutation enables us to describe the Cutmix sample as a concatenation of cropped vectors: $[\mathbf{v}^{i,\prime}_{:\sum \mathbf{M}}; \mathbf{v}^{j,\prime}_{\sum \mathbf{M}+1:}]$. Because shuffling, cropping, and concatenation conserve variance, the Cutmix sample exhibits the same variance as that of $\mathbf{v}$. The same goes for the mean.

Furthermore, numerous variants of Cutmix have been proposed, which have modified the sampling of binary masks $\mathbf{M}$ for a more natural combination of images (Kim, Choo, and Song 2020; Uddin et al. 2021; Walawalkar et al. 2020; Huang, Wang, and Tao 2021). We find that all these Cutmix variants still follow the masked combination scheme $\bar{\mathbf{v}} = \mathbf{M} \odot \mathbf{v}^i + (\mathbf{1} - \mathbf{M}) \odot \mathbf{v}^j$, which conserves variance. In summary, Cutmix and its variants conserve the variance of the input image, which prevents and is safe from the variance shift in positional embedding. However, we claim that the use of Mixup should be reconsidered for training ViT because it causes variance shifts. Note that we are not saying Mixup is wrong; rather, our claim is that Mixup has a side effect of variance shift, which may outweigh the advantage of training combined samples.

Variance Shift Due to Random Erasing

Random erasing (Zhong et al. 2020), also referred to as Cutout (Devries and Taylor 2017), is a widely adopted data augmentation technique. Random erasing randomly selects and drops a certain block from the input image during training. The original study of random erasing proposed several versions for erasing behavior, such as replacing the block with zero (`const` mode), replacing the block with a single random normal value (`rand` mode), and replacing the block with a per-pixel random normal value (`pixel` mode). Again, to investigate the variance shift in positional embedding, we examine the variance of the image after random erasing. Our findings are as follows:

Proposition 3 *Random erasing conserves variance if we configure it to* `pixel` *mode and adopt the correct mean-std normalization using dataset statistics.*

In fact, replacing a certain block of the input image is similar to the Cutmix scheme. For a vector $\mathbf{v}$, random erasing selects a binary mask $\mathbf{M}$ where each element is either zero or one and replaces the corresponding block with that of the erasing vector $\mathbf{e}$, which yields a random erasing sample $\hat{\mathbf{v}} = \mathbf{M} \odot \mathbf{v} + (\mathbf{1} - \mathbf{M}) \odot \mathbf{e}$. Similar to the Cutmix scheme, when investigating variance, we may permute the order of the vector to describe it as concatenation: $[\mathbf{v}'_{:\sum \mathbf{M}}; \mathbf{e}'_{\sum \mathbf{M}+1:}]$. The property of the concatenated vector can be examined through Lemma 1 in the Appendix, which says that the concatenated vector achieves $\mathrm{E}[\hat{\mathbf{v}}] = \mathrm{E}[\mathbf{v}]$ and $\mathrm{Var}[\hat{\mathbf{v}}] = \mathrm{Var}[\mathbf{v}]$ if $\mathrm{E}[\mathbf{v}] = \mathrm{E}[\mathbf{e}]$ and $\mathrm{Var}[\mathbf{v}] = \mathrm{Var}[\mathbf{e}]$. This property holds for an arbitrary drop probability of random erasing.

Here, the `const` mode provides an erasing vector as a zero vector $\mathbf{e} = \mathbf{0}$, the `rand` mode samples a single value $e_i \sim \mathcal{N}(0, 1)$ shared across all elements, and the `pixel` mode independently samples each element $e_i \sim \mathcal{N}(0, 1)$. Thus, in `const` and `rand` modes, the erasing vector $\mathbf{e}$ comprises a single value, which exhibits $\mathrm{Var}[\mathbf{e}] = 0$ and thereby causes $\mathrm{Var}[\hat{\mathbf{v}}] \neq \mathrm{Var}[\mathbf{v}]$. Only the `pixel` mode has different elements that are sampled from $\mathcal{N}(0, 1)$, which leads to $\mathrm{Var}[\mathbf{e}] = 1$. Therefore, to maintain the consistency of mean and variance after random erasing, we should ensure the `pixel` mode, $\mathrm{E}[\mathbf{v}] = 0$, and $\mathrm{Var}[\mathbf{v}] = 1$.

Fortunately, the conditions $\mathrm{E}[\mathbf{v}] = 0$ and $\mathrm{Var}[\mathbf{v}] = 1$ tend to hold for standard training recipes because applying global mean-std normalization using dataset statistics has been standard practice. When training ViT on the ImageNet dataset, researchers have used mean-std normalization with pre-computed ImageNet statistics of per-channel mean $(0.485, 0.456, 0.406)$ and standard deviation (std) $(0.229, 0.224, 0.225)$. These statistics are referred to as default-mean-std, which guarantees $\mathrm{E}[\mathbf{v}] = 0$ and $\mathrm{Var}[\mathbf{v}] = 1$ for the input images and ensures consistent variance after random erasing with `pixel` mode.

However, several exceptions violate this condition. A prime example is so-called inception-mean-std (Szegedy et al. 2015; Steiner et al. 2022), which applies mean-std normalization using per-channel mean $(0.5, 0.5, 0.5)$ and std $(0.5, 0.5, 0.5)$. Adopting these values causes $\mathrm{E}[\mathbf{v}] \neq 0$ and $\mathrm{Var}[\mathbf{v}] \neq 1$ and thereby inconsistent mean and variance after random erasing, even when using the `pixel` mode. In fact, the original implementation of ViT (Dosovitskiy et al. 2021) preferred inception-mean-std. Note that the original study of ViT used a much larger dataset of JFT-300M with few data augmentations that excluded random erasing, which is safe from variance shifts due to random erasing. However, other follow-up studies have trained ViT and its variants using the ImageNet dataset with rich data augmentations including random erasing, which is vulnerable to the inception-mean-std. Because using random erasing is a standard practice for training ViT, we claim that we should configure it as `pixel` mode and simultaneously adopt correct mean-std normalization using the default-mean-std, avoiding the inception-mean-std.

We find that the use of inception-mean-std in the original implementation of ViT has influenced other follow-up studies on variants of ViT. For example, the original study of BEiT (Bao et al. 2022) used inception-mean-std, whereas BEiT v2 (Peng et al. 2022) opted for the default-mean-std using ImageNet statistics. For certain models of MaxViT (Tu et al. 2022), inception-mean-std was used, whereas for other models, default-mean-std was adopted. Furthermore, the pretrained ViTs provided by MIIL (Ridnik et al. 2021) simply used per-channel mean $(0, 0, 0)$ and std $(1, 1, 1)$. Considering these practices where the default-mean-std is unused but random erasing is used in the training of ViT, we claim that a variance shift arises, and therefore we should opt for the correct mode of random erasing as well as the correct mean-std normalization using dataset statistics.

Experiments

In this section, we conduct one-by-one verification on findings from the previous section through experiments.

Rescaling Positional Embedding

ImageNet Evaluation We examine the effect of rescaling positional embedding when using upsampled images dur-

Size	ViT-S/16 224^2			ViT-B/16 224^2			ViT-L/16 224^2		
	Baseline	**Ours**	**Diff**	**Baseline**	**Ours**	**Diff**	**Baseline**	**Ours**	**Diff**
224^2	81.386	81.296	-0.090	84.528	84.344	-0.184	85.834	85.570	-0.264
288^2	82.024	82.044	+0.020	84.810	84.846	+0.036	86.288	86.382	+0.094
352^2	81.438	81.470	+0.032	84.204	84.406	+0.202	85.762	85.788	+0.026
416^2	80.008	80.320	+0.312	83.056	83.320	+0.264	84.944	85.026	+0.082
480^2	77.928	78.472	+0.544	81.538	82.100	+0.562	84.034	84.158	+0.124
544^2	75.424	76.296	+0.872	79.374	80.422	+1.048	82.868	82.924	+0.056
608^2	72.384	73.722	+1.338	76.744	78.384	+1.640	81.492	81.590	+0.098
672^2	68.964	70.882	+1.918	73.560	76.048	+2.488	79.982	80.106	+0.124
736^2	65.014	67.704	+2.690	69.832	73.324	+3.492	78.494	78.654	+0.160
800^2	60.954	64.196	+3.242	65.674	70.524	+4.850	76.800	77.096	+0.296
864^2	56.528	60.710	+4.182	61.330	67.280	+5.950	75.282	75.656	+0.374

Table 3: Top-1 accuracy (%) for ImageNet with respect to various spatial sizes. Baseline indicates the existing practice of using $\mathrm{UP}(\mathbf{P})$, and ours indicates calibrating the variance using $\mathrm{UP}(\mathbf{P})/\sqrt{k}$.

ing the test phase. We target ViTs pretrained ImageNet, provided by AugReg (Steiner et al. 2022) that pretrained ViTs on ImageNet-21K and fine-tuned on ImageNet-1K with 224×224 size. We examined ViT-$\{$S, B, L$\}$ with a patch size of 16, where S, B, and L stand for small, base, and large models, respectively. For these models, we examined the test size across 224×224 to 864×864. For each test size, we first evaluated the top-1 accuracy on the validation set of ImageNet using the existing practice of upsampling positional embedding $\mathrm{UP}(\mathbf{P})$, which corresponds to the baseline performance. Subsequently, we performed the same evaluation procedures with rescaled positional embedding $\mathrm{UP}(\mathbf{P})/\sqrt{k}$, which corresponds to the proposed method to prevent variance shift.

Table 3 summarizes the results. Firstly, upsampling to larger sizes incurs a distribution shift of facing different images, which yields decreased top-1 accuracy. However, we discovered that the decreased performance is not only caused by different images but also by the variance shift of positional embedding: Rescaling the positional embedding removed the hidden factor of degraded performance and recovered the top-1 accuracy to a certain degree. Indeed, our analysis says that when using a test size larger than 224×224, we should rescale positional embedding. The results were exactly in agreement with our claim: Rescaling the positional embedding improved top-1 accuracy when using a test size larger than 224×224 but degraded performance when using a test size of 224×224, which corresponds to the case that does not require upsampling the positional embedding. Note that certain resolutions such as 288×288 yielded improved accuracy for models trained with 224×224. Generally, a larger resolution of the test image causes a dataset shift, which may degrade performance. However, a larger resolution image may contain rich image information, which helps image understanding and thereby leads to performance gain. We conjecture that this phenomenon explains the slight performance gains.

Semantic Segmentation Now we examine rescaling the positional embedding targeting semantic segmentation tasks. The ADE20K dataset (Zhou et al. 2017) contains scene-centric images along with the corresponding segmentation labels. A crop size of 512×512 pixels was used, which was obtained after applying mean-std normalization and a random resize operation using a resize scale of (2048, 512) pixels with a ratio range of 0.5 to 2.0. Furthermore, a random flipping with a probability of 0.5 and the photometric distortions were applied. The objective was to classify each pixel into one of the 150 categories and train the segmentation network using the pixel-wise cross-entropy loss. The same goes for the LoveDA dataset (Wang et al. 2021) with 7 categories and the Cityscapes dataset (Cordts et al. 2016) with 19 categories, except for using a crop size of 1024×1024 pixels for the Cityscapes dataset. Note that the resize operation applies 1D upsampling or 2D upsampling depending on the dataset; 2D upsampling is applied for the LoveDA dataset, whereas 1D upsampling is applied for the ADE20K and Cityscapes datasets. The rescaling values $1/\sqrt{k}$ were referred to in Table 2.

We used DeiT-S/16 (Touvron et al. 2021a) pretrained on ImageNet and UPerNet (Xiao et al. 2018) with a multi-level neck. For the ADE20K dataset, pretrained segmentation networks were downloaded; for the LoveDA and Cityscapes datasets, we trained segmentation networks by ourselves. To follow common practice for semantic segmentation, training recipes from `MMSegmentation` were employed (Contributors 2020). For training, AdamW optimizer with weight decay 10^{-2}, betas $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate 6×10^{-5} with polynomial decay of the 160K scheduler after linear warmup were used. The training was performed on a 4×A100 GPU machine.

We measured three indices commonly used in semantic segmentation—all pixel accuracy (aAcc), mean accuracy of each class (mAcc), and mean intersection over union (mIoU) (Table 4). The baseline corresponds to using $\mathrm{UP}(\mathbf{P})$, whereas the proposed method corresponds to $\mathrm{UP}(\mathbf{P})/\sqrt{k}$. We observed that for all datasets and indices,

Dataset	Index	$\mathrm{UP}(\mathbf{P})$	$\mathrm{UP}(\mathbf{P})/\sqrt{k}$	Diff
LoveDA	aAcc	65.58	66.46	+0.88
	mIoU	47.78	48.37	+0.59
	mAcc	62.00	62.20	+0.20
ADE20K	aAcc	80.73	80.74	+0.01
	mIoU	43.16	43.27	+0.11
	mAcc	54.28	54.36	+0.08
Cityscapes	aAcc	96.02	96.04	+0.02
	mIoU	77.20	77.31	+0.11
	mAcc	84.27	84.40	+0.13

Table 4: Results (%) on semantic segmentation before and after applying $1/\sqrt{k}$ rescaling to positional embedding. 2D upsampling is applied for the LoveDA dataset, whereas 1D upsampling is applied to the ADE20K and Cityscapes datasets; this choice depends on the rectangular or square image size. The proposed rescaling consistently improved the performance.

Mean-Std	Data Augmentation	ViT-S/16	ViT-B/16
Default	Mixup 0.0, Cutmix 0.0	74.801	74.741
	Mixup 0.0, Cutmix 1.0	**79.540**	**80.240**
	Mixup 0.2, Cutmix 1.0	78.069	79.448
	Mixup 0.4, Cutmix 1.0	78.128	79.248

Table 5: Top-1 accuracy (%) for ImageNet. Cutmix improves performance, whereas Mixup does not.

rescaling the positional embedding consistently improved the segmentation performance.

Effect of Mixup and Cutmix

ImageNet Training We examine the effect of Mixup and Cutmix in training ViTs. In contrast to previous experiments, we now train ViTs ourselves. The ImageNet dataset contains 1.28M images for 1,000 classes. For the image classification experiments with ImageNet, we used the pytorch-image-models library, also known as `timm` (Wightman 2019). We referred to the hyperparameter recipe described in the official documentation and the recipe from DeiT. For training, AdamW optimizer (Loshchilov and Hutter 2019) with learning rate 5×10^{-4}, epochs 300, warm-up learning rate 10^{-6}, cosine annealing schedule (Loshchilov and Hutter 2017), weight decay 0.05, label smoothing (Szegedy et al. 2016) 0.1, RandAugment (Cubuk et al. 2020) of magnitude 9 and noise-std 0.5 with increased severity (rand-m9-mstd0.5-inc1), stochastic depth (Huang et al. 2016) 0.1, mini-batch size 288 per GPU, Exponential Moving Average of model weights with decay factor 0.99996, and image resolution 224×224 were used. The training was performed on an 8×A100 GPU machine, which requires from two to three days per training.

Table 5 summarizes the results. Comparing the results without and with Cutmix, we find that Cutmix is significantly beneficial to performance. However, when additionally applying Mixup, the top-1 accuracy rather decreased, which indicates that the side effect of variance shift outweighs the advantage of training combined samples of Mixup. These results are in agreement with our claim: Cutmix is suitable for training ViT, whereas Mixup is not owing to variance shift.

Mean-Std	Data Augmentation	ViT-S/16	ViT-B/16
Inception	RE 0.00 with `pixel` mode	**79.426**	**80.134**
	RE 0.25 with `pixel` mode	79.126	79.894
	RE 0.50 with `pixel` mode	78.726	79.794
Default	RE 0.25 with `pixel` mode	**79.158**	**80.415**
	RE 0.25 with `rand` mode	79.009	80.398
	RE 0.25 with `const` mode	78.909	80.298

Table 6: Top-1 accuracy (%) for ImageNet. When using random erasing (RE), its `pixel` mode and default-mean-std configuration are required.

Configurations of Random Erasing

Now, we examine the effect of configuration on random erasing using different probabilities and modes. We used the same recipe as before, but without Mixup and with Cutmix.

Table 6 summarizes the result. When using inception-mean-std, applying random erasing rather decreased the top-1 accuracy, and the recipes without random erasing worked better. In contrast, when using default-mean-std, the best result was found for the `pixel` mode, whereas `rand` and `const` modes yielded slightly lower performance. All these results validate our claim: When applying random erasing, we should choose the default-mean-std and `pixel` mode.

Conclusion

This study reported the variance shift in the positional embedding of ViTs caused by data augmentations. We discussed four data augmentations: random resize crop, Mixup, Cutmix, and random erasing. Firstly, we showed that common upsampling techniques lead to inconsistent variance and described a resultant variance shift in positional embedding. Furthermore, we identified that Mixup incurs variance shifts, whereas Cutmix is safe from this issue. We further analyzed the detailed behavior of random erasing to reveal its correct configuration for variance consistency—`pixel` mode with mean-std normalization using dataset statistics. The proposed methods were validated through various experiments on ViTs, where removing the variance shift in positional embedding consistently improved the performance of ViTs.

Acknowledgments

This work was supported by Samsung Electronics Co., Ltd (IO201210-08019-01).

References

Ali, A.; Touvron, H.; Caron, M.; Bojanowski, P.; Douze, M.; Joulin, A.; Laptev, I.; Neverova, N.; Synnaeve, G.; Ver-

beek, J.; and Jégou, H. 2021. XCiT: Cross-Covariance Image Transformers. In *NeurIPS*, 20014–20027.

Ba, L. J.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. *CoRR*, abs/1607.06450.

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2022. BEiT: BERT Pre-Training of Image Transformers. In *ICLR*.

Chen, C. R.; Fan, Q.; and Panda, R. 2021. CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification. In *ICCV*, 347–356.

Contributors, M. 2020. MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark. https://github.com/open-mmlab/mmsegmentation.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 3213–3223.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. 2020. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In *NeurIPS*.

d'Ascoli, S.; Touvron, H.; Leavitt, M. L.; Morcos, A. S.; Biroli, G.; and Sagun, L. 2021. ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases. In *ICML*, volume 139, 2286–2296.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Devries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. *CoRR*, abs/1708.04552.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

Fang, Y.; Wang, W.; Xie, B.; Sun, Q.; Wu, L.; Wang, X.; Huang, T.; Wang, X.; and Cao, Y. 2023. EVA: Exploring the Limits of Masked Visual Representation Learning at Scale. In *CVPR*, 19358–19369.

Han, K.; Xiao, A.; Wu, E.; Guo, J.; Xu, C.; and Wang, Y. 2021. Transformer in Transformer. In *NeurIPS*, 15908–15919.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. B. 2022. Masked Autoencoders Are Scalable Vision Learners. In *CVPR*, 15979–15988.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Heo, B.; Yun, S.; Han, D.; Chun, S.; Choe, J.; and Oh, S. J. 2021. Rethinking Spatial Dimensions of Vision Transformers. In *ICCV*, 11916–11925.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep Networks with Stochastic Depth. In *ECCV (4)*, volume 9908, 646–661.

Huang, S.; Wang, X.; and Tao, D. 2021. SnapMix: Semantically Proportional Mixing for Augmenting Fine-grained Data. In *AAAI*, 1628–1636.

Jiang, Z.; Hou, Q.; Yuan, L.; Zhou, D.; Shi, Y.; Jin, X.; Wang, A.; and Feng, J. 2021. All Tokens Matter: Token Labeling for Training Better Vision Transformers. In *NeurIPS*, 18590–18602.

Kim, B. J.; Choi, H.; Jang, H.; Lee, D. G.; Jeong, W.; and Kim, S. W. 2023. Improved robustness of vision transformers via prelayernorm in patch embedding. *Pattern Recognit.*, 141: 109659.

Kim, B. J.; and Kim, S. W. 2024. Scale Equalization for Multi-Level Feature Fusion. *CoRR*, abs/2402.01149.

Kim, J.; Choo, W.; and Song, H. O. 2020. Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup. In *ICML*, volume 119, 5275–5285.

Liu, Z.; Hu, H.; Lin, Y.; Yao, Z.; Xie, Z.; Wei, Y.; Ning, J.; Cao, Y.; Zhang, Z.; Dong, L.; Wei, F.; and Guo, B. 2022. Swin Transformer V2: Scaling Up Capacity and Resolution. In *CVPR*, 11999–12009.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*, 9992–10002.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *ICLR*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *ICLR*.

Peng, Z.; Dong, L.; Bao, H.; Ye, Q.; and Wei, F. 2022. BEiT v2: Masked Image Modeling with Vector-Quantized Visual Tokenizers. *CoRR*, abs/2208.06366.

Ridnik, T.; Baruch, E. B.; Noy, A.; and Zelnik, L. 2021. ImageNet-21K Pretraining for the Masses. In *NeurIPS Datasets and Benchmarks*.

Ryali, C.; Hu, Y.; Bolya, D.; Wei, C.; Fan, H.; Huang, P.; Aggarwal, V.; Chowdhury, A.; Poursaeed, O.; Hoffman, J.; Malik, J.; Li, Y.; and Feichtenhofer, C. 2023. Hiera: A Hierarchical Vision Transformer without the Bells-and-Whistles. In *ICML*, volume 202, 29441–29454.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Steiner, A.; Kolesnikov, A.; Zhai, X.; Wightman, R.; Uszkoreit, J.; and Beyer, L. 2022. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers. *Trans. Mach. Learn. Res.*, 2022.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*, 1–9.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2818–2826.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021a. Training data-efficient image transformers & distillation through attention. In *ICML*, volume 139, 10347–10357.

Touvron, H.; Cord, M.; and Jégou, H. 2022. DeiT III: Revenge of the ViT. In *ECCV (24)*, volume 13684, 516–533.

Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; and Jégou, H. 2021b. Going deeper with Image Transformers. In *ICCV*, 32–42.

Tu, Z.; Talebi, H.; Zhang, H.; Yang, F.; Milanfar, P.; Bovik, A. C.; and Li, Y. 2022. MaxViT: Multi-axis Vision Transformer. In *ECCV (24)*, volume 13684, 459–479.

Uddin, A. F. M. S.; Monira, M. S.; Shin, W.; Chung, T.; and Bae, S. 2021. SaliencyMix: A Saliency Guided Data Augmentation Strategy for Better Regularization. In *ICLR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*, 5998–6008.

Walawalkar, D.; Shen, Z.; Liu, Z.; and Savvides, M. 2020. Attentive Cutmix: An Enhanced Data Augmentation Approach for Deep Learning Based Image Classification. In *ICASSP*, 3642–3646.

Wang, J.; Zheng, Z.; Ma, A.; Lu, X.; and Zhong, Y. 2021. LoveDA: A Remote Sensing Land-Cover Dataset for Domain Adaptive Semantic Segmentation. In *NeurIPS Datasets and Benchmarks*.

Wightman, R. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models.

Xiao, T.; Liu, Y.; Zhou, B.; Jiang, Y.; and Sun, J. 2018. Unified Perceptual Parsing for Scene Understanding. In *ECCV (5)*, volume 11209, 432–448.

Yun, S.; Han, D.; Chun, S.; Oh, S. J.; Yoo, Y.; and Choe, J. 2019. CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In *ICCV*, 6022–6031.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2020. Random Erasing Data Augmentation. In *AAAI*, 13001–13008.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene Parsing through ADE20K Dataset. In *CVPR*, 5122–5130.